```cpp
// process_text_stream.cpp
//
#ifdef _WIN32
    #pragma warning(disable:4996)
    #include <tchar.h>
    #include <windows.h>
    #include <conio.h>
    #include <direct.h>
#elif (defined __linux__) || (defined _AIX) || (defined __APPLE__)
    #include <stdlib.h>
    #include <sys/types.h>
    #include <sys/stat.h>
    #include <unistd.h>
    typedef char _TCHAR;
    #define _tmain main
#endif

#include <stdio.h>
#include <string.h>
#include <fstream>
#include <iostream>
#include <iomanip>
using namespace std;

#include "Veta.h"

#define MAXLINE 255

void my_getch();
void GetNamesOfIOFiles(const char[], char[]);
//---------------------------------------------------------------------------
int _tmain(int argc, _TCHAR* argv[])
{
    ifstream in;
    VetaClass** veta;
    unsigned i, j, k, n, PocetViet;
    char c, * str, nameoffile[MAXLINE];

    cout << "\n  Tento program nacita vetu z textoveho suboru, vyhodi z nej "
        << "prebytocne\nmedzery a na obrazovku vypise upravenu vetu a potom"
        << " pod seba jednotlive slova\nspolu s ich dlzkami.\n\n";

    GetNamesOfIOFiles("VETY.TXT", nameoffile);
    in.open(nameoffile, ios::in);
    if (!in) {
        cout << "  Subor VETY.TXT sa nepodarilo otvorit.\n";
        my_getch();
        exit(1);
    }

    i = 0;
    veta = new VetaClass * [50];
    do {
        do {
            in.get(c);
        } while ((!isalnum(c)) && (!ispunct(c)) && (!in.eof()));

        if ((isalnum(c)) || (ispunct(c))) {
            veta[i] = new VetaClass;
            in.putback(c);
            in >> veta[i];
            i++;
        }
    } while (!in.eof());
    PocetViet = i;
    in.close();

    cout << "  Zadana veta:\n\n";
    cout << "  ";
    for (i = 0; i < PocetViet; i++)
        cout << veta[i];
    cout << "\n\n  Press any key!";
    my_getch();
```

```
74
75      n = 0;
76      cout << "\n\n  Slova vety           Dlzka slov\n";
77      for (i = 0; i < PocetViet; i++) {
78          for (j = 0; j < veta[i]->PocetSlov; j++) {
79              str = veta[i]->DajSlovo(j);
80              k = (unsigned)strlen(str);
81              cout << "\n  " << str;
82              cout.setf(ios::right);
83              cout << setw(27 - k) << k;
84              n++;
85              if (n % 40 == 0) {
86                  n = 0;
87                  cout << "\n\n  Press any key!\n";
88                  my_getch();
89              }
90          }
91      }
92      cout << endl;
93
94      for (i = 0; i < PocetViet; i++)
95          delete veta[i];
96      delete[] veta;
97
98      my_getch();
99      return 0;
100 }
101 //---------------------------------------------------------------------------
102 void my_getch()
103 {
104 #ifdef _WIN32
105     _getch();
106 #else
107     cout << endl;
108 #endif
109 }
110 //---------------------------------------------------------------------------
111 void GetNamesOfIOFiles(const char name_of_input_file[], char path_to_input_file[])
112 {
113     char current_path[MAXLINE];
114     current_path[0] = '\0';
115
116 #ifdef _WIN32
117     TCHAR exePath[MAXLINE];
118
119     HMODULE hModule = GetModuleHandle(NULL);
120     if (hModule != NULL) {
121         if (!GetModuleFileName(hModule, exePath, MAXLINE)) {
122             cout << "Nepodarila sa zistit cesta k exe-suboru.\n";
123             my_getch();
124             exit(1);
125         }
126     }
127     else {
128         cout << "Module handle is NULL.\n" << endl;
129         my_getch();
130         exit(1);
131     }
132 #ifdef __BORLANDC__
133     int iii;
134     bool flag = false;
135     for (iii = strlen(exePath); iii >= 0; iii--) {
136         if (!flag && exePath[iii] == '\\') {
137             current_path[iii + 1] = '\0';
138             flag = true;
139         }
140         if (flag)
141             current_path[iii] = (char)exePath[iii];
142     }
143 #else
144     int iii;
145     bool flag = false;
146     for (iii = (int)wcslen(exePath); iii >= 0; iii--) {
```

```
147             if (!flag && exePath[iii] == '\\') {
148                 current_path[iii + 1] = '\0';
149                 flag = true;
150             }
151             if (flag)
152                 current_path[iii] = (char)exePath[iii];
153         }
154     #endif
155     #elif (defined __linux__) || (defined __APPLE__)
156         unsigned iii;
157         char line[MAXLINE];
158         FILE* fp;
159         if ((fp = popen("/bin/pwd", "r")) == NULL) {
160             perror("popen error");
161             exit(1);
162         }
163         if (fgets(line, MAXLINE, fp) == NULL) {
164             perror("fgets error");
165             exit(1);
166         }
167         pclose(fp);
168 
169         iii = 0;
170         while (line[iii] != '\r' && line[iii] != '\n') {
171             current_path[iii] = line[iii];
172             iii++;
173         }
174         current_path[iii] = '\0';
175     #elif (defined _AIX)
176         unsigned iii;
177         char line[MAXLINE];
178         FILE* fp;
179         if ((fp = popen("user/bin/pwd", "r")) == NULL) {
180             perror("popen error");
181             exit(1);
182         }
183         if (fgets(line, MAXLINE, fp) == NULL) {
184             perror("fgets error");
185             exit(1);
186         }
187         pclose(fp);
188 
189         iii = 0;
190         while (line[iii] != '\r' && line[iii] != '\n') {
191             current_path[iii] = line[iii];
192             iii++;
193         }
194         current_path[iii] = '\0';
195     #endif
196 
197         path_to_input_file[0] = '\0';
198         strcat(path_to_input_file, current_path);
199     #if (defined __linux__) || (defined _AIX) || (defined __APPLE__)
200         strcat(path_to_input_file, "/inputs/");
201     #elif (defined _WIN32)
202         strcat(path_to_input_file, "inputs\\");
203     #endif
204         strcat(path_to_input_file, name_of_input_file);
205     }
206     //--------------------------------------------------------------------------
207 
```